Appendix

```
;**********************************************************************
;
;
;           File Name: MAIN.S
;
;       7/25/00         - SN V0.1
;
;       This is the main module. When not processing anything the execution remains
;       in this module at all times.
;
;**********************************************************************
;
;       must be in this order
;


        .include            "fvt.inc"
        .include            "keydef.inc"
        .include            "data.inc"
        .include            "equ.inc"
        .include            "macro.inc"

        .global KEY_NUMBER, KEY_NUMBER_BUFFER

        .extern irq0
        .extern irq1
        .extern irq2
        .extern irq3
        .extern irq4
        .extern irq5


        .extern         check_key
        .extern         ScanKeyPad
        .extern         WaitForKeyPress
        .extern         ServiceCode
        .extern         wait_key_off
        .extern         set_t16_timer
        .extern         disable_t16_timer
        .extern         DialOut
        .extern         delay_100ms
        .extern         SWInitialize

;***********************************************************
;       interrupt vectors
;***********************************************************
```

```
;
irq0_vec:
      .word  irq0
irq1_vec:
      .word  irq1
irq2_vec:
      .word  irq2
irq3_vec:
      .word  irq3
irq4_vec:
      .word  irq4
irq5_vec:
      .word  irq5
;*********************************************************
;
;
;      RESET - main entry
;
;*********************************************************
;
;*********************************************************
; Initialize stack pointer hi and low bytes.
;*********************************************************

      ld          SPH,  #HIGH(stack_ptr)
      ld          SPL,  #LOW (stack_ptr)

;*********************************************************
;
;      Initialize port modes.
;      p01m and p2m defaults are used. They are fine.
;*********************************************************

      ld          P3M, #00h            ; p32, p33, p31, p30 = inputs
      ld          P2M, #0ffh           ; p2 = inputs (for keyscan rows)
      ld          P01M, #00000100b        ; p0 = outputs (for leds) and keyscan
columns

;*********************************************************
; Reset the counters.
;*********************************************************

      srp    EXTEND_GROUP_D
      ld          ctr0, #T8_RESET_TOUT
      ld          ctr1, #03h      ;(#TC8_16_OUT)| (#INIT_T8_OUT_HIGH)|
(#INIT_T16_OUT_HIGH)        ; port 3.6 is timer output
    ld          ctr2, #T16_RESET_TOUT

    srp   EXTEND_GROUP_F
    ld          wdtmr,#00
    ld          0, #0FEh            ;PCON register is 0

; ********* Strictly for OTP 'E72/E73 **************
    ld          0eh, #00000100b


;
;
; BRRRR... Start me up..

;** check if warm or cold start **
    tbitnz smr, #BIT7, warm      ;Check for warm start.
;

    ;** COLD START **
    ld          smr, #00100000b

    ;** Clear Internal RAM if COLD START and RAM CURROPTED **
    srp   REG_GROUP
    cp          CHECK1, #0AAh        ;Check if RAM should be
cleared after cold start
    jr          ne, ClearRam
    cp          CHECK2, #0AAh
    jr          ne, ClearRam
    cp          CHECK3, #0AAh
    jr          eq, warm

ClearRam:
    srp   0
    ld          r5, SPL

ClearLoop:
    clr   @r5
    dec   r5
    cp          r5,#05h
    jr          nz,ClearLoop

    ld          CHECK1, #0AAh        ;load fields with values to be
checked on cold boot
    ld          CHECK2, #0AAh
```

```
	ld		CHECK3, #0AAh


;***********************************************************
;	Warm start
;	check key depressed
;	if key is not depressed then return with
;	KEY_NUMBER = 0xff;
;***********************************************************
warm:
	srp	REG_GROUP
	ld		MODE,#0
	ld		P_MODE,#0
	ld		p2,#0fh ;initialize p2
	ld		p1,#47h ;Initialize p1
	or		p3,#BIT6	;CS=high
	ld		p0,#BIT3	;RS0=0

	RedLedOff		;Make sure the record led is off
	ld		r1,p3
	or		p3,#BIT1
	or		p3,#BIT2
	tbitz	p3,#BIT1,DialOutBaby		;Check Battery Voltage
	tbitz	p3,#BIT2,DialOutBaby		;Check Battery Voltage


	call	WaitForKeyPress		;Wait for key press
	jr		nc, warm			; None, do other stuff..

GotANewValidKey:
	call	ServiceCode			;serice the key accordingly.
	call	WaitForKeyPress		;wait for another key.
	jr		c, GotANewValidKey
	call	disable_t16_timer	;disable timers.

CheckforBatteryVoltage:

	tbitz	p3,#BIT1,DialOutBaby
	tbitz	p3,#BIT2,DialOutBaby

	jr		warm			; If both batteries OK.
					;continue on.
DialOutBaby:
```

```
;Check battery voltage if voltage is less
                                                  ;than
normal. Dial out.
;       ld      41h, #02h
;       ld      42h,#08h
;       ld      43h,#02h
;       ld      44h,#09h
;       ld      45h,#0ffh
;       ld      46h, #09h
;       ld      47h,#03h
;       ld      48h,#06h
;       ld      49h,#05h
;       ld      4ah, #03h
;       ld      4bh,#05h
        ld      4ch,#0ffh

        call            delay_100ms
        call    DialOut
        jr      warm                            ; go back to
work....

        .byte "Copyright (c)2000-2001 Chamberlain Group. Developed by Yamtech Inc, 847
963 2829"
```

```
;************************************************************x
;    SERVICE KEY
;            File Name: SRVKEY.S
;
;        8/14/00 - V1.0 SN
;        This routine services the keys on the charger unit.
;        The main functions provided are
;        1. learning the phone number.
;        2. Enable recording of ogm.
;        3. Enable playback of ogm.
;
;
;************************************************************

    .include  "fvt.inc"
    .include  "keydef.inc"
    .include  "data.inc"
    .include  "equ.inc"
    .include  "macro.inc"

    .global ServiceCode
    .global DeviceLightsOff
    .global DevLightKey


    .extern FlashGreenLed
    .extern WaitForKeyReleaseFlashRed
    .extern wait_key_off, port_delay
    .extern WaitForKeyRelease,WaitForKeyPressUserDelay, WaitForKeyPress
    .extern WaitForKeyReleaseAndStartThreeSecTimer
    .extern Delay70ms, set_t16_timer, disable_t16_timer
;************************************************************
;    Service key code
;    Key Number in 'KEY_NUMBER'
;************************************************************
ServiceCode:
    cp    KEY_NUMBER, #MAX_VALID_KEY
    jr    ugt, ServiceExit              ;check if key pressed is valid


    cp    KEY_NUMBER, #KEY_RECORD
    jp    eq, ProcessRecordKey              ;Record OGM

    cp         KEY_NUMBER,#KEY_PROGRAM
```

```
     jp          eq,LearnAPhoneNumber    ;ProcessProgramKey
                ;Program phone number


;    call   WaitForKeyReleaseAndStartThreeSecTimer ;if device key pressed for

;    jp     nc, ProcessDeviceKey             ;three seconds learn a code
;    ld     KEY_FOR_CODE_FLAG, KEY_NUMBER
;    jp     LearnAPhoneNumber

ServiceExit:
     ret

;*********************************************************
;
; LearnAPhoneNumber
;
;
;    Function:
;         Learns the Phone number. Max of 10 digits. The first
;         digit cannot be a 0.
;
;    Inputs:
;         DEVICE_FLAG
;    Returns:
;         CF = 0  - OK          Set PhoneNumberOK Flag
;         CF = 1  - Error       Reset PhoneNumberOK Flag
;
;    Modifies:
DigitPointer   .equ   40h    ;Store Digits
LoopCounter    .equ   r8
Multiplier     .equ   r11
CodeEntered    .equ   r13
;
;    Subords:
;         WaitForKeyPress
;         CheckFirstDigit
;         mul_8
;
;*********************************************************
LearnAPhoneNumber:

     or          MODE, #PROGRAM_MODE
     and          p1,#10111111b
```

```
        ld          DigitPointer,#41h
  continue:
        call    wait_key_off          ;Wait for release of key
        ldrr    LOOP_COUNTER_H,LOOP_COUNTER_L,T30_SECONDS
        call    WaitForKeyPressUserDelay       ;waits 30 seconds in learn mode
        jp          nc,LearnPhoneError

  CheckForDigit:
        call    CheckFirstDigit              ;  check if first digit is valid
        jp      c, LearnPhoneError           ;
        ld          @DigitPointer, KEY_NUMBER       ;first digit OK store it
        inc     DigitPointer
  StoreDigits:
        call    WaitForKeyPress              ;loop to get more digits or
  time out
        jp          nc, LearnPhoneError
        GreenLedOn
        cp          KEY_NUMBER, #9              ;if number key not
  pressed exit error
        jp          ugt, LearnPhoneError          ;key pressed is not a digit, exi
  t
  error

        ld          @DigitPointer, KEY_NUMBER       ;load the digit into storage
        inc     DigitPointer                 ;increment pointer to digits

        djnz    LoopCounter, StoreDigits       ;decrement digit counter

  LearnPhoneOK:
        ld          @DigitPointer,#0ffh           ;Indicates termination of
  phone number
        RedLedOff                            ;exit point when things are good
        ld      MODE, #0
        rcf
        or          p1,#40h
        or          P_MODE,#PhoneNumberOK   ;Set phonenumberOK Flg
        call    wait_key_off                 ;wait for key release
        ret

  LearnPhoneError:                                ;exit point when things are bad
        ld          @DigitPointer,#0ffh           ;Indicates termination of
  phone number
        RedLedOff
        or          p1,#40h
```

```
      ld    MODE, #0
      and    P_MODE,#~PhoneNumberOK  ;reset PhoneNumberOk Flg
      scf
      ret


;*************************************************************************
; CheckFirstDigit
;
;
;     Function:
;           Checks the 1st digit of an entered code. It must not be zero.
;           If the first digit is one the learn nine more digits or else learn
;           six more digits.
;
;     Returns:
;           CF = 0        - 1st digit is valid for the device being learned.
;           CF = 1        - 1st digit is invalid ...
;
;     Modifies:
;
;     LoopCounter:
;
;     Subords:
;           None.
;
;*************************************************************************
CheckFirstDigit:
      ld          LoopCounter,#10              ;Default 10 digits
      cp           KEY_NUMBER, #KEY_0
      jr          eq,FirstWrong

      cp           KEY_NUMBER, #KEY_1      ;If first # !=1 then enter only 7
digits
      jr          eq,FirstOK

      ld          LoopCounter,#6

FirstOK:
      rcf
      ret

FirstWrong:
      ld          LoopCounter,#0         ;Enable Playback if Zero key
```

```
is pressed first
      scf
      jr          ProcessPlayBack
      ret

;*************************************************************
; ProcessRecordKey:
;
; Enables Recording of Out Going Message
;
;*****************************************************
ProcessRecordKey:
      call   wait_key_off         ;Wait for release of key
      ldrr   LOOP_COUNTER_H,LOOP_COUNTER_L,T30_SECONDS      ;20
second max recording
                                          ;but
wait for 30 seconds just in case.
      ld          p1,#06h              ;Turn on recording.
      call   port_delay
      call   WaitForKeyPressUserDelay      ;waits 30 seconds in learn mode
      jp          nc,RecordError
      ld          p1,#07h       ;Turn off recording
RecordError:
      ld          p1,#07h       ;Turn off recording
      or     MODE,#0h              ;Recording completed.
      ret

ProcessPlayBack:              ;Program + 0 will initiate Playback
      call   wait_key_off
      ld          p1,#03h
      call   port_delay
      ldrr   LOOP_COUNTER_H,LOOP_COUNTER_L,T30_SECONDS      ;20
second max recording
                                          ;but
wait for 30 seconds just in case.
      call   WaitForKeyPressUserDelay      ;waits 30 seconds in learn mode
      jp          nc,RecordError

      ld          p1,#07h       ;Turn off recording
      scf                        ;set carry flag to exit from
program mode.
      jr          RecordError
```

```
Srvkey

.end
```

```
;*******************************************************************************
; FILENAME: keynew.src

;
;

;    DESCRIPTION:

;    Parses the key pad. There are 12 keys on the key pad.
;    The program key allows the user to program the phone number to dial.
;    The record key allows the user to record the outgoing message.
;

;    REVISION HISTORY:

;         Version: 0.1
;         Date: 07/25/00, Author:
;
;
;

; *** Public Functions ***

;

;

;

; ** Internal Functions **

;
```

```
;
;*************************************************************************
;




;** include files **
      .include "fvt.inc"
      .include "data.inc"
      .include "equ.inc"
      .include "keydef.inc"
      .include  "macro.inc"

;** external functions **
      .extern set_t16_timer              ; Sets up timer for keyscan time-out.
      .extern disable_t16_timer
      .extern mul_8
      .extern FlashGreenLed
      .extern FlashRedLed
      ;.extern InterDigitDelay
;** public functions **
      .global delay_500uS
      .global delay_100ms
      .global WaitForKeyReleaseAndStartThreeSecTimer
      .global WaitForKeyReleaseFlashRed
      .global WaitForKeyRelease
      .global WaitForKeyPress
      .global WaitForKeyPressUserDelay
      .global ScanKeyPad
      .global wait_key_off
      .global delay_10ms
      .global delay_ms
      .global check_key
      .global port_delay

;*******************************************************
;     Wait for key depressed or
;     timeout if no key in 15 seconds
;
;     check every 32 msec.
;     return with cf =1 & key_number
;*******************************************************
WaitForKeyPress:
      tbitnz  MODE, #PROGRAM_MODE, SetActionTime         ;Time
```

```
out for Program mode
      ldrr   LOOP_COUNTER_H,LOOP_COUNTER_L,T8_SECONDS

      jr     NotActionTime

SetActionTime:
      ldrr   LOOP_COUNTER_H,LOOP_COUNTER_L,T8_SECONDS

NotActionTime:
      call   wait_key_off                ;if key already pressed, wait
here

      tbitz  MODE,#PROGRAM_MODE, NoRedLED
      GreenLedOn

NoRedLED:

WaitForKeyPressUserDelay:
      ldw
CURR_LOOP_COUNTER_H,CURR_LOOP_COUNTER_L,LOOP_COUNTER_H,LOOP_CO
UNTER_L
      or     IO_FLAGS,#TIMEOUT_FLAG          ; This enables the time-out
routine in IRIRQ.S
      call   set_t16_timer               ; Give them a limited time in
which to respond

WaitForKeyOrTimeOut:
      call   ScanKeyPad
      jr     nc, NoKey_CheckForTimeOut

      ld     tmp_key, KEY_NUMBER
      ld     loop_cnt,#50

 bounce_lop:
      call   ScanKeyPad
      jr     nc, NoKey_CheckForTimeOut       ; no key
      cp     tmp_key, KEY_NUMBER
      jr     nz, NoKey_CheckForTimeOut       ; not matched
      djnz   loop_cnt,bounce_lop

      call   disable_t16_timer             ; Found a key, Stop the Time_out
 counter.
      scf
      ret
```

```
NoKey_CheckForTimeOut:
     tbitnz  IO_FLAGS,#TIMEOUT_FLAG, WaitForKeyOrTimeOut

     call   disable_t16_timer            ; Timed_out, Stop the Time_out counter.
     rcf
     ret

;sole control key matrix
;*************************************************************
;     Scan Key Pad
;
;   set key matrix ports for the SC500 series
;
;          col 0 = p0.0   row 0 = p2.0
;          col 1 = p0.1   row 1 = p2.1
;          col 2 = p0.2   row 2 = p2.2
;                         row 3 = p2.3
;
;     if key is not depressed then return with
;     KEY_NUMBER = 0xff;
;*************************************************************
ScanKeyPad:
     and    p0, #KeyPadMask              ; All columns are low
     ld     KEY_NUMBER,#0ffh               ;default key number
;
;*************************************************************
;    Get row #, scan ports p2.0 to p2.3
;*************************************************************
;
     clr    row                    ;set row counter to zero
     ld     i,#00000001b                ;start bit at pin 2.0
row_scan_loop:
     cp     row,#4
     jr     ugt,exit_key_scan
     ld     j,i
     and    j,p2                   ;compare mask to port 2
     jr     z,row_found                ;zero flag is set if row is found
     inc    row                    ;increment row counter
     rl     i                   ;try next row
     jr     nc,row_scan_loop             ;do while c f is not set, end loop after
 8
rows
     jr     exit_key_scan
row_found:
```

```
;
;*********************************************************
;
;     Get col #, scan ports p0.0 to p0.2
;*********************************************************
;
;
      clr    col                    ;set col counter to 0
      or     p0, #11111111b              ; Start with Column0 (p0.0)
      and    p0, #11111110b               ; clr p0.0
col_scan_loop:
      cp     col,#4
      jr     ugt,exit_key_scan

      call   port_delay              ;allow port to settle
      call   port_delay              ;allow port to settle, maybe you donot
need two delays but leave itfor now.

;     cp     p2,#0fh              ;check if a row pin gets pulled low
      ld     r5,p2
      and    r5,#0fh
      cp     r5,#0fh
      jr     ne, compute_key_num           ;if a row pin is pulled low exit loop
      ld     r5, p0
      or     r5, #KeyPadMask
      rl     r5
      or     r5, #KeyPadMask
      and    p0, r5
      jr     nc,exit_key_scan            ;if pins 2 thru 7 were checked exit loop
      inc    col                    ;else increment col counter
      jr     col_scan_loop              ;begin loop again, check next
column

exit_key_scan:
      and    p0,#KeyPadMask                    ;reset the key
output ports
      rcf                           ;no valid key pressed
      ret                           ;return 0xff in
KEY_NUMBER
;
;*********************************************************
;
;     Compute key number and return it
;     Formula: row * 3 + col = key number
;*********************************************************
;
compute_key_num:
      ld     r11, row
      ld    r13, #3
      call  mul_8
      add   r13, col
      ld    KEY_NUMBER, r13
      and   p0, #KeyPadMask            ; All columns are low
      call  TranslateKeyNumber
      scf
      ret                         ;return valid key number in
KEY_NUMBER

WaitForKeyRelease:
wait_key_off:
      and   p0, #KeyPadMask            ; All columns are low
off_lp:
      call  check_key
      jr    c, off_lp
      ret

WaitForKeyReleaseFlashRed:
      and   p0, #KeyPadMask            ; All columns are low
off_lp0:
      call  check_key
      jr    c, off_lp0
      ret

;
;*************************************************************
;     Miscellaneous Delay Routines
;*************************************************************
delay_10ms:
      push  i
      ld    i, #20
delay_10ms_loop:
      call  delay_500uS
      djnz  i, delay_10ms_loop
      pop   i
      ret

delay_100ms:
      push  r4
      ld          r4,#50
delay_100ms_loop:
      call  delay_10ms
      djnz  r4,delay_100ms_loop
```

```
        pop         r4
        ret

    delay_500uS:
        push    i
        ld      i,#23
    d148uS:
        call    port_delay
        djnz    i, d148uS
        pop     i
        ret
    delay_ms:
    BlinkOne:
        GreenLedOn
        call    delay_500uS

    BlinkOneHereToo:
        GreenLedOff

        call    delay_500uS
        djnz    i, delay_ms
        ret

    ;*********************************************************
    ;   Delay 108usec + 40 usec for the call
    ;*********************************************************
    port_delay:
        push    j
        push    j
        pop     j
        pop     j

        ret

    ;*********************************************************
    ;   Check key ON(true), OFF(false)
    ;   return:
    ;           cf = 1 if key depressed
    ;           cf = 0 if no key
    ;*********************************************************
    check_key:
        and     p0, #KeyPadMask
    ;   cp      p2, #0ffh
        ld      r0,p2
```

```
	and	r0,#0fh		;P00-P02 should be high.
	cp	r0,#0fh
	jr	eq, NoKeyIsPressed

	scf
	ret

NoKeyIsPressed:
	rcf
	ret

;*********************************************************
;	Wait For Key Release And Start Three Second Timer
;	    Determines if key was pressed for 3 seconds.
Timer		.equ	r0
;*********************************************************
WaitForKeyReleaseAndStartThreeSecTimer:
	ld	Timer,#0ffh

KeepTiming:
	dec	Timer
	jr	z, ThreeSecPassed
	call	delay_10ms
	call	check_key
	jr	c, KeepTiming
	ret

ThreeSecPassed:
	scf
	ret

;*****************************************************************
;*
;* Translate key number
;*
;*****************************************************************
TranslateKeyNumber:
	ldrr	r0, r1, Translate
	addw	r0,r1,#0,KEY_NUMBER
	ldc	r2,@rr0
	ld	KEY_NUMBER, r2
	ret

TransError:
        ld      KEY_NUMBER, #0ffh
        ret

Translate:
        .byte   KEY_1
        .byte   KEY_2
        .byte   KEY_3
        .byte   KEY_4
        .byte   KEY_5
        .byte   KEY_6
        .byte   KEY_7
        .byte   KEY_8
        .byte   KEY_9
        .byte   KEY_RECORD
        .byte   KEY_0
    .byte       KEY_PROGRAM
        .end
```

```
;*********************************************************
;       file name: irutil.s
;                       7/27/93
;       utility modules
;
;*********************************************************
;

        .include  "keydef.inc"
        .include  "fvt.inc"
        .include  "data.inc"
        .include  "equ.inc"
        .include  "macro.inc"


        .global mul_8
        .global mult_16
        .global set_t16_timer
        .global disable_t16_timer
        .global FlashGreenLed
        .global FlashRedLed

        .extern delay_10ms

;*********************************************************
;       Init timer 16 counter
;       set clock/8(each tick - 2 usec)
;       Terminal counts = 128 msec.
;*********************************************************
set_t16_timer:
        push    rp

        srp 2dh ;REG_GROUP + EXTEND_GROUP_D
        ld      tc16l,#0ffh
        ld      tc16h,#0ffh
        ld      ctr2,#26h       ;T16_CLK_2MHZ+T16_RESET_TOUT+T16_ENA_INT
; enable interrupt
        ld      ctr1, #11110011b        ; Set to normal mode
        or      ctr2,#T16_ENABLE
        or      0fbh, #MSK_3
        ei

        pop     rp
        ret
```

```
;***********************************************************
;     Disable timer 16 counter
;     set clock/8(each tick - 2 usec)
;     Terminal counts = 128 msec.
;***********************************************************
disable_t16_timer:
      push   rp

      srp    2dh     ;REG_GROUP + EXTEND_GROUP_D

      ld     ctr2, #T16_RESET_TOUT
      and    0fbh, #~ MSK_3

      pop    rp

      and    IO_FLAGS,#~ TIMEOUT_FLAG
      ret

;***********************************************************
;
;     FlashGreenled
;
;
;
;***********************************************************
FlashGreenLed:
      push   r8
      ld     r8,#10
fl_10:
      GreenLedOn
      call   delay_10ms
      GreenLedOff
      call   delay_10ms
      call   delay_10ms
      djnz   r8, fl_10
      pop    r8
      ret

;***********************************************************
;
;     FlashRedled
;
```

```
;
;
;*********************************************************
FlashRedLed:
      push   r8
      ld     r8,#5
frl_10:
      RedLedOn
      call   delay_10ms
      call   delay_10ms
      RedLedOff
      call   delay_10ms
      call   delay_10ms
      djnz   r8, frl_10
      pop    r8
      ret
;*********************************************************
;     Perform a 8 bit by 8 bit unsigned binary multiplication
;     input:  r11  = 8 multiplier
;             r12  = 0
;             r13  = 8 multiplicand
;     return:
;             rr12=  product
;
;*********************************************************
mul_8:
      ld     mul_LEN, #9
      clr    product_HI
      rcf
lp1:
      rrc    product_HI
      rrc    product_LO
      jr     nc,nxt1
      add    product_HI,MULTIPLIER
nxt1:
      djnz   mul_LEN,lp1
      ret

;*********************************************************
; Function:
;     mult_16
; multiply 16 bit number n number of times
;
;     r9-> # of times
;       r10->h byte of the multiplicand
;       r11->l byte of the multiplicand
;       rr12->subordinates
;
;**********************************************************

mult_16:
        ld      r12,r10
        ld      r13,r11
        dec     r9
        jr      z,m_16ret
m_16:
        addw    r10,r11,r12,r13
        djnz    r9,m_16
m_16ret:
        ret
```

```
;*********************************************************************
;
;        File Name: DIALOUT.asm
;
;    8/25/00       - SN V0.1
;
;    This function does the dialing out to the phone line and piping the audio signal
;
;    Inputs: none
;
;    Outputs: none
;
;*********************************************************************
;
;    must be in this order
;

     .include          "fvt.inc"
     .include          "keydef.inc"
     .include          "data.inc"
     .include          "equ.inc"
     .include          "macro.inc"

     .global      DialOut

     .extern      port_delay
     .extern      enable_t16_timer
     .extern      disable_t16_timer
     .extern      set_t16_timer
     .extern      delay_100ms
     .extern      delay_10ms

DigitPointer   .equ    40h

DialOut:

;Port 0 inoutput mode only. We donot read the DTMF signals, in.

;To dial out
;Initialize XECOM
;pull OH High
;/WR = Low
;/RD = High
;/CS = Low
```

```
;D//V = Low
;D4-D1= Digit transmitted.
;Wait till /RI goes high, Indicates the ring is stopped.
;Pipe the audio signal
;Hang up and exit
        ld              p1,#87h         ;turn on grn light
        call        InitXecom

        ld              DigitPointer,#41h
        or              p1,#10000000b           ;P1.7 pull high,
OH, off hook
        call        delay_100ms
        call        delay_100ms
KeepDialing:
        ld              r2,@DigitPointer
        cp              r2,#0ffh
        jr              eq, DialingDone
;       and              p2,#C_BIT3
;       ld              p3,#00100000b


;       or              p1,#10000000b           ;P1.7 pull high,
OH, off hook
        call        delay_100ms

        call        GetDigit
;       and              p0,#C_BIT3
;       ld              p3,#00100000b

;       call   delay_100ms
        OutDReg         r2
;       and              p0,#C_BIT3
        ld              p3,#00100000b

;       ld    p3,#00010000b


;*      ld              p3,#01010000b
        ld              p3,#01000000b
;       call        delay_100ms
        or              p0,#BIT3
;       ld              p3,#00010000b
;       ld              p3,#01000000b
```

```
	inc		DigitPointer
	jr		KeepDialing


;	or		p3,#BIT6		;Chip Select in inactive
;Supposedly connected
;enable playback for 30 seconds...
DialingDone:
	ld		p3,#01100000b
	call	delay_100ms
	call	delay_100ms
;	jr		HangUp		;***
DialingDone1:
;	call	port_delay
;	tbitz	p2,#BIT4,DialingDone1	; Test for Ring Indicator

	ldrr
CURR_LOOP_COUNTER_H,CURR_LOOP_COUNTER_L,T8_SECONDS	;20 second max
recording
	or		IO_FLAGS,#TIMEOUT_FLAG		; This
enables the time-out routine in IRIRQ.S
	call	set_t16_timer		; Give them a
limited time in which to respond

	call	port_delay
WaitForTimeOut1:
	tbitnz IO_FLAGS,#TIMEOUT_FLAG, WaitForTimeOut1

LoopHere:
	call	port_delay
	ldrr
CURR_LOOP_COUNTER_H,CURR_LOOP_COUNTER_L,T30_SECONDS	;20 second max
recording
	or		IO_FLAGS,#TIMEOUT_FLAG		; This
enables the time-out routine in IRIRQ.S
	call	set_t16_timer		; Give them a
limited time in which to respond
	and		p1,#0fbh
	; Turn on the Audio pipe
	call	port_delay
WaitForTimeOut:
	tbitnz IO_FLAGS,#TIMEOUT_FLAG, WaitForTimeOut
	call	disable_t16_timer		; Timed_out, Stop the
Time_out counter.
```

```
     ld                 p1,#87h
     ; Turn off the audio pipe


     call          port_delay
     ldrr
CURR_LOOP_COUNTER_H,CURR_LOOP_COUNTER_L,T30_SECONDS     ;20 second max
recording
     or                 IO_FLAGS,#TIMEOUT_FLAG          ; This
enables the time-out routine in IRIRQ.S
     call          set_t16_timer                 ; Give them a
limited time in which to respond
     ld                 p1,#083h
     ; Turn on the Audio pipe
     call          port_delay
WaitForTimeOut0:
     tbitnz  IO_FLAGS,#TIMEOUT_FLAG, WaitForTimeOut0
     call          disable_t16_timer             ; Timed_out, Stop the
Time_out counter.
     ld                 p1,#0f7h
     ; Turn off the audio pipe



HangUp:
     and     p1,#C_BIT7                  ;Hang up and out of here
ret
     ret

WriteDTMF:
;Sets up XECOM for writing to DTMF port
;    ld            P01M, #00000100b       ; p0 = outputs (for leds) and keyscan
columns


     and           p0,#C_BIT3            ;RS0=0
     call    port_delay
     and           p3,#C_BIT4            ;/WR=0
     or            p3,#BIT5            ;/RD=1
     and           p3,#C_BIT6            ;CS=0
     ret

ReadStatus:
     ld     p01m,#01000100b
```

```
        and   p3,#C_BIT6          ;cs=0
        or    p0,#BIT3
        or    p3,#BIT4
        and   p3,#C_BIT5
        call  delay_10ms
        nop
        nop
        ret


InitXecom:
;       ld    p0,#BIT3       ;RS0=1
;*      ld    p3,#00010000b

;*      ld    p3,#00100000b
        ld    p3,#01010000b
        ld    p0,#19h;18h    ;38h          ;RegA,NoInterrupt,DTMF
mode,Touch Tone mode
        ld    p3,#00100000b
;       ld    p3,#00100000b  ;Toggle line
;       ld    p0,#18h ;38h

;       ld    p3,#00100000b
        ld    p3,#01010000b
        ld    p0,#99h;98h;98h        ; Write to register B. Burst mode transmit

        ld    p3,#00100000b
        ld    p3,#01010000b
        ld    p0,#08h ;08h

        ld    p3,#00100000b  ;*
;       ld    p0,#08h          ;*

;       ld    p3,#00100000b

        ld    p3,#01010000b
;       or    p0,#08h
;       ld    p3,#00010000b
;       ld    p3,#01010000b
        ret


        ret
```

```
GetDigit:
      ldrr   r0, r1, TranslateDigits
      addw   r0,r1,#0,r2
      ldc    r2,@rr0

      ret

TransError:
      ld     r2, #0ffh
      ret
TranslateDigits:
            .byte        DIGIT0
            .byte        DIGIT1
            .byte        DIGIT2
            .byte        DIGIT3
            .byte        DIGIT4
            .byte        DIGIT5
            .byte        DIGIT6
            .byte        DIGIT7
            .byte        DIGIT8
            .byte        DIGIT9

DATA0  .equ   00000000b
DIGIT1 .equ   00010001b
DIGIT2 .equ   00100000b
DIGIT3 .equ   00110001b
DIGIT4 .equ   01000000b
DIGIT5 .equ   01010001b
DIGIT6 .equ   01100000b
DIGIT7 .equ   01110001b
DIGIT8 .equ   10000000b
DIGIT9 .equ   10010001b
DIGIT0 .equ   10100000b
```

```
;*************************************************************
;     INTERRUPT SERVICE MODULES
;
;                File Name: IRQ.asm
;
;
;*************************************************************

     .include "fvt.inc"
     .include "keydef.inc"
     .include "data.inc"
     .include "equ.inc"
     .include "macro.inc"

;** external functions **


;*************************************************************
;     UNUSED Interrupts
;*************************************************************

     .global irq0
     .global irq1
     .global irq2
     .global irq3
     .global irq4
     .global irq5


;*************************************************************
;
;     IRQ3
;     if io_flags.KEY_FLAG then decr. loop_count
;     if loop counts == 0 then stop count down
;
;*************************************************************
irq3:
     push   rp
;
     srp    OUT_GROUP+EXTEND_GROUP_D
     tm     ctr2,#T16_RESET_TOUT
     jr     z,irq3_ret
;
     or     ctr2,#T16_RESET_TOUT
```

```
;
;       Decr. Loop Counter
;
        tm      io_flags,#TIMEOUT_FLAG
        jr      z,irq3_20       ; no flag
;
        subw    curr_loop_counter_h,curr_loop_counter_l,#0,#1
        jr      nz,irq3_ret     ;not yet time out
; check low
        cp      curr_loop_counter_l,#0
        jr      nz,irq3_ret     ;not yet time out
; Reset
        and     io_flags,#~ TIMEOUT_FLAG
        jr      irq3_ret
;
;       Other functions
;
irq3_20:
;
;       return
;
irq3_ret:
        pop     rp
        iret
```

```
;*********************************************************
;       UNUSED
;*********************************************************
irq0:
irq1:
irq2:
irq4:
irq5:
        iret

;*********************************************************
        .end
```

```
;       .list off
;************************************************************
;
;     KEY Assignment for Battery Charger
;                   keydef.h
;                   8/03/00
;
; So far we have the # pad and two extra keys.
;************************************************************
BIT_0         .equ   0
BIT_1         .equ   1
BIT_2         .equ   2
BIT_3         .equ   3
BIT_4         .equ   4
BIT_5         .equ   5
BIT_6         .equ   6
BIT_7         .equ   7

; key number

KEY_0          .equ   0
KEY_1          .equ   1
KEY_2          .equ   2
KEY_3          .equ   3
KEY_4          .equ   4
KEY_5          .equ   5
KEY_6          .equ   6
KEY_7          .equ   7
KEY_8          .equ   8
KEY_9          .equ   9
KEY_RECORD     .equ   10
KEY_TEST          .equ   11
KEY_PROGRAM          .equ   12
MAX_VALID_KEY  .equ   12          ;last valid key

;************************************************************
```

```
     .list off
;**********************************************************
;     file name: data.h
;
;**********************************************************


; unit = 32 mseconds

T0_SECONDS     .equu        0
T2_SECONDS     .equ        55
T4_SECONDS     .equ   122
T5_SECONDS     .equ   4500/32
T8_SECONDS     .equ   270
T15_SECONDS     .equ   15000/32
T30_SECONDS     .equ   30000/32
T60_SECONDS .equ      60000/32

;utility flags
FLAG1          .equ   BIT0
FLAG2          .equ   BIT1
FLAG3          .equ   BIT2
FLAG4          .equ   BIT3
FLAG5          .equ   BIT4
FLAG6          .equ   BIT5
FLAG7          .equ   BIT6
FLAG8          .equ   BIT7


;**********************************************************
;     internal ram/reg allocation
;**********************************************************
;
;     Register group 0 & 1 for stack ptr
;     Start from reg4
;
;**********************************************************
stack_ptr      .equ   0efh
;**********************************************************
;     Utility Group 0 - start from 4
;     0,1,2,3 are IO port
;**********************************************************
UTL_GROUP           .equ   00h
;
```

```
;
;**************************************************************
;       register group 1
;       Holds a 16-bit pointer to a DAT for a given device
;       (AUX) can be any device.
;
;**************************************************************
PERM_GROUP          .equu   10h



; *******************************************************************
MODE                .equ    11h
PROGRAM_MODE   .equ    BIT1


; *******************************************************************
P_MODE              .equ    12h     ;default -> repeat everything

PhoneNumberOK       .equ    BIT1

;**************************************************************
;       register group 2
;       used general purpose register group
;**************************************************************
REG_GROUP           .equ    20h

col                 .equ    r0
row                 .equ    r1

i                   .equ    r3
j                   .equ    r4

loop_cnt            .equ    r9
tmp_key             .equ    r7
;**************************************************************
;       Register group 3 - FOR OUTPUT MODULE
;**************************************************************
OUT_GROUP           .equ    30h

io_flags            .equ    r0
IO_FLAGS            .equ    30h
```

```
TIMEOUT_FLAG          .equ   BIT1

LOOP_COUNTER          .equ   31H
LOOP_COUNTER_L        .equ   31H
LOOP_COUNTER_H        .equ   32H

curr_loop_counter     .equ   r3
curr_loop_counter_l   .equ   r3
curr_loop_counter_h   .equ   r4
CURR_LOOP_COUNTER     .equ   33H
CURR_LOOP_COUNTER_L   .equ   33H
CURR_LOOP_COUNTER_H   .equ   34H

KEY_FOR_CODE_FLAG     .equ   39h
TEMP_MODE    .equ   3ah


KEY_NUMBER            .equ   3ch

KEY_NUMBER_BUFFER     .equ   3eh
CHECK1                .equ   3fh     ;leave this here

;***********************************************************
;      Register Group 40h
; This register group holds the phone number, one digit per byte.
;
;***********************************************************
TEL_DAT_POINTER_GROUP        .equ   40h




CHECK2                .equ   6fh     ;leave this here because Vince
wants it here

;***********************************************************
;
;***********************************************************
;SPARE REG BANK       0

CHECK3                .equ   7fh
SPARE_GROUP           .equ   090h
```

```
COUNTER             .equ   090h

HIGH_CODE           .equ   091h
LOW_CODE            .equ   092h


MISC_FLAGS          .equ   094h         ; MISC_FLAGS
SleepFlag        .equ   BIT2        ; MISC_FLAGS

CHECK4            .equ   98h


FIRST_DIGIT         .equ   9ah
SECOND_DIGIT          .equ   9bh
THIRD_DIGIT         .equ   9ch

timer_high        .equ   r14
timer_low         .equ   r15

TIMER_HIGH          .equ   9eh
TIMER_LOW           .equ   9fh


END_OUT_REGS         .equ   9fh

;*************************************************************
;
;*************************************************************
;
TEMP_X      .equ   0d0h
TEMP_Y      .equ   0d1h


;*************************************************************
;     Control Registers
;*************************************************************
CTRL_GROUP          .equ   0f0h
;*************************************************************
;     extend data group
;*************************************************************
EXTEND_GROUP_D       .equ   0dh
EXTEND_GROUP_F       .equ   0fh
;*************************************************************
      .list on
```

```
;*************************************************************
;      misc equates
;*************************************************************
mul_LEN        .equ   r14
MULTIPLIER     .equ   r11
product_LO     .equ   r13
product_HI     .equ   r12
```

09955374.062801

```
        .list off
;*************************************************************
;       file name: fvt.h
;
*************
;       Output & Edge detector
;
;       p31 - edge detector
;       p34 - t8_out
;       p35 - t8_out & t16_out logic
;       p36 - t16_out
;
;*************************************************************
;
;       GENERAL EQUATES
;
BIT0    .equ    01h
BIT1    .equ    02h
BIT2    .equ    04h
BIT3    .equ    08h
BIT4    .equ    10h
BIT5    .equ    20h
BIT6    .equ    40h
BIT7    .equ    80h

C_BIT0  .equ    11111110b
C_BIT1  .equ    11111101b
C_BIT2  .equ    11111011b
C_BIT3  .equ    11110111b
C_BIT4  .equ    11101111b
C_BIT5  .equ    11011111b
C_BIT6  .equ    10111111b
C_BIT7  .equ    01111111b
;
;       PORTS
;
Port0   .equ    00
Port1   .equ    01
Port2   .equ    02
Port3   .equ    03

;
;
;       register definitions
```

```
;
;    17x registers
;
;    bank D
;
ctr0   .equ   00h
ctr1   .equ   01h
ctr2   .equ   02h
tc8l   .equ   04h
tc8h    .equ   05h
tc16l   .equ   06h
tc16h   .equ   07h
lo16    .equ   08h
hi16    .equ   09h
lo8    .equ   0ah
hi8    .equ   0bh
;
;    Bank F
;
;
pcon    .equ   00     ;xxxx xxx0
smr          .equ   0b    ;0010 00x0
smr2   .equ   0d     ;x0x0 00xx
wdtmr  .equ   0f     ;xxx0 1101
;
;    Control register 0
;    Counter/timer 8 control register
;
T8_ENABLE      .equ   BIT7
;
T8_SINGLE      .equ   BIT6
T8_RESET_TOUT  .equ   BIT5   ;reset flag to 0
T8_CLK_4MHZ    .equ   00
T8_CLK_2MHZ    .equ   BIT3
T8_CLK_1MHZ    .equ   BIT4
T8_CLK_1_2MHZ  .equ   BIT4+BIT3

T8_ENA_INT     .equ   BIT1   ;enable Time-out int.(IRQ3)
P34_OUT       .equ   BIT0

T16_ENABLE     .equ   BIT7
T16_ENABLE_C   .equ   07fh

T16_SINGLE     .equ   BIT6   ;transmit mode
```

```
T16_IGNORE_EDGE .equ    BIT6    ;t16 ignore edge
T16_RESET_TOUT  .equ    BIT5    ;reset flag to 0
T16_CLK_4MHZ    .equ    00
T16_CLK_2MHZ    .equ    BIT3
T16_CLK_1MHZ    .equ    BIT4
T16_CLK_1_2MHZ  .equ    BIT4+BIT3

T16_CAP_INT     .equ    BIT2    ;enable data capture int.

T16_ENA_INT     .equ    BIT1    ;enable Time-out int.(IRQ3)
P35_OUT         .equ    BIT0
;
;       delay unit based on t16 - use 2mhz
;       terminal counts = 32 mseconds
;
UNIT            .equ    010000h/2/1000
;
;       piem - port 1 mode selection regiser
;
piem            .equ    0ch
P1_ADDR         .equ    BIT4
P1_H_IMPEDENCE  .equ    BIT4+BIT3
;
;       pcon - port configuration register
;
;
P36_P00_COMPR   .equ    BIT0
;
;       p3m - port 3 mode register
;

P2_PUSH_PULL    .equ    BIT0

P31_ANALOG_MODE .equ    BIT1


P33_IN_P34_OUT  .equ    00
P33_IN_P34_DM   .equ    BIT3
P33_DV_P34_RDY  .equ    BIT4

P31_DV_P36_RDY  .equ    BIT5            ;TOUT
;
;       p01m
;       p0 & P1 - port 0 & 1 mode register
```

```
;
; p00 - p03 mode
;
P00_OUT       .equ   00
P00_IN       .equ   BIT0
P00_ADDR       .equ   BIT1


STACK_INTERNAL .equ   BIT2

P01_OUT       .equ   00
P01_IN       .equ   BIT3
P01_ADDR       .equ   BIT4
P01_HIMPE      .equ   BIT4+BIT3


EXT_MEM_EXTEND .equ   BIT5
;
; p04 - p07 mode
;
P04_OUT       .equ   00
P04_IN       .equ   BIT6
P04_ADDR       .equ   BIT7
;
;     ipr - interrupt priority reg
;
;     IRQ - interrupt request reg
;
IRQ_0        .equ   BIT0   ;P32 input
IRQ_1        .equ   BIT1   ;P33 input
IRQ_2        .equ   BIT2   ;P31 input
IRQ_3        .equ   BIT3   ;TC16 output/TC16 timeout
IRQ_4        .equ   BIT4   ;TC8 output/TC8 timeout
IRQ_P31L_P32L  .equ   00
IRQ_P31L_P32H  .equ   BIT6
IRQ_P31H_P32L  .equ   BIT7
IRQ_P31H_P32H  .equ   BIT7+BIT6
;
;     msk - interrupt mask reg.
;
MSK_0        .equ   BIT0   ;P32 input(enable)
MSK_1        .equ   BIT1   ;P33 input
MSK_2        .equ   BIT2   ;P31 input
MSK_3        .equ   BIT3   ;TC16 output/TC16 timeout
```

```
MSK_4          .equu    BIT4    ;TC8 output/TC8 timeout
MSK_P31L_P32L  .equ    00
MSK_P31L_P32H  .equ    BIT6
MSK_P31H_P32L  .equ    BIT7
MSK_P31H_P32H  .equ    BIT7+BIT6
        .list on
        .list off
;*************************************************************
;       file name: equ.h
;
;*************************************************************
TRUE          .equ   1
FALSE         .equ   0

ON            .equ   1
OFF           .equ   0

YES           .equ   1
NO            .equ   0

HIGH   .equ   1
LOW           .equ   0

ACTIVE_LOW          .equ   0
ACTIVE_HIGH         .equ   BIT1
BIT_COMPLEMENT         .equ   BIT2

;** Status LED's - p00, p01 **
GreenLedEnable  .equ    C_BIT1
GreenLedDisable .equ    BIT1

RecordLedEnable .equ    C_BIT0
RecordLedDisable       .equ    BIT0


KeyPadMask      .equ    0f8h
```

```
;       .list off
;*****************************************************************
srp     .macro  arg1
;       .byte   31h,#(arg1)
        ld      rp,#arg1
        .endm
;*****************************************************************
;       STOP macro
;*****************************************************************
m_stop  .macro
        ei
        nop
        nop
        stop
        .endm
;*****************************************************************
;       HALT macro
;*****************************************************************
m_halt  .macro
        ei
        nop
        nop
        halt
        .endm

Select_Xecom    .macro
        and     p3,#C_BIT6
        .endm

ToggleBits      .macro
        and     p0,#0fh                 ;reset
        ld      p3,#00000000b
;       and     p3,#C_BIT4                  ;Reset p3.4
;       and     p3,#C_BIT5                  ;Reset p3.5
        and     p0,#C_BIT3                  ;Reset p0.3 RS0
;       or      p3,#BIT6                ;CS=1

        .endm

;*****************************************************************
;       rs0wrrd macro
;*****************************************************************

rs0wrrd .macro  const1,const2,const3
```

```
    call  port_delay
    or    p3,#C_BIT6              ;CS=0
    or          p0,#const1            ;Set p0.3 appropriately
;   and          p3,#const2            ;set P3.4
;   or          p3,#const3            ;set P3.5
    ld          p3,#00100000b       ;rd=1, wr=0
    call  port_delay
    .endm

;***********************************************************
;   outdata macro
;***********************************************************
outdata .macro  const1
;   and          p0,#00001111b
    ld          p0,#const1
    .endm
;***********************************************************
;   OutDReg macro
;   Same as outdata but uses register
;***********************************************************

OutDReg .macro reg
;   and    p0,#00001111b
    ld          p0,reg

    .endm

;***********************************************************
;   turn on green led

;***********************************************************
GreenLedOn     .macro
    and    p0, #GreenLedEnable
    .endm
;***********************************************************
;   turn off green led

;***********************************************************
GreenLedOff    .macro
    or     p0, #GreenLedDisable
    .endm
;***********************************************************
;   turn on red led
```

```
;**********************************************************
RedLedOn      .macro
      and    p0, #00h       ;RedLedEnable
      .endm
;**********************************************************
;     turn off red led

;**********************************************************
RedLedOff     .macro
      or     p0, #0ffh      ;RedLedDisable
      .endm

;**********************************************************
;     load pair register
;     reg1 = high, reg2 = low
;     reg3 = high, reg4 = low
;**********************************************************
ldw    .macro  reg1,reg2,reg3,reg4
      ld     reg1,reg3
      ld     reg2,reg4
      .endm

;**********************************************************
;     Load pair register
;     reg1 = high reg
;     reg2 = low reg
;**********************************************************
ldrr   .macro  reg1,reg2,const
      ld     reg1, #HIGH(const)
      ld     reg2, #LOW(const)
      .endm

;**********************************************************
;     add a word
;     tgtlow,tgthigh = result
;     srclow,srchigh = adder
;**********************************************************
addw    .macro tgthgh,tgtlow,srchgh,srclow
      add    tgtlow,srclow
      adc    tgthgh,srchgh
      .endm

;**********************************************************
;     subtract a word
;     tgtlow,tgthigh = result
;     srclow,srchigh = adder
;************************************************************
subw   .macro tgthgh,tgtlow,srchgh,srclow
      sub    tgtlow,srclow
      sbc    tgthgh,srchgh
      .endm

;************************************************************
;     subtract a word
;     tgtlow,tgthigh = result
;     srclow,srchigh = adder
;************************************************************
sub3byte      .macro tgthgh,tgtmid, tgtlow,srchgh,srcmid,srclow
      sub    tgtlow,srclow
      sbc    tgtmid,srcmid
      sbc    tgthgh,srchgh
      .endm

;************************************************************
;     Shift to right through carry
;************************************************************
shtr   .macro reg0
      rcf
      rrc    reg0
      .endm
;************************************************************
;     Shift to left through carry
;************************************************************
shtl   .macro reg0
      rcf
      rlc    reg0
      .endm
;************************************************************
;     Test bit and jump if zero flag = 1
;************************************************************
tbitz  .macro flag,bit,jmp
      tm     flag,bit
      jr     z,jmp
      .endm
;************************************************************
;     Test bit and jump if zero flag = 0
;************************************************************
tbitnz .macro flag,bit,jmp
```

```
        tm      flag,bit
        jr      nz,jmp
        .endm
;**********************************************************
        .list on
```

```
Zilog Linkage Editor.  Version T2.11    16-May-101    18:15:11    Page:  1

LINK MAP:

Date:     Wed May 16 18:15:11 2001
Processor: Z8
Files:   [Command] D:\battcharger\dcbc.cmd
         [Object ] D:\battcharger\main.o
         [Object ] D:\battcharger\KEYNEW.o
         [Object ] D:\battcharger\SRVKEY.o
         [Object ] D:\battcharger\Dialout.o
         [Object ] D:\battcharger\UTIL.o
         [Object ] D:\battcharger\irq.o

COMMAND LIST:
=============
 1: -q D:\battcharger\dcbc.cmd
 2: ; ZDS Generated Linker Command File
 3: -A
 4: -g
 5: -m "D:\battcharger\dcbc.map"
 6: Range RFILE %0,%100
 7: Range XDATA %4000,%C000
 8: Range ROM %0,%4000
 9: -o "D:\battcharger\dcbc"
10: "D:\battcharger\main.o"
11: "D:\battcharger\KEYNEW.o"
12: "D:\battcharger\SRVKEY.o"
13: "D:\battcharger\Dialout.o"
14: "D:\battcharger\UTIL.o"
15: "D:\battcharger\irq.o"
 Zilog Linkage Editor.  Version T2.11    16-May-101    18:15:11    Page:  2

SPACE ALLOCATION:
=================

Space                          Base     Top      Span
------------------------------ -------- -------- --------
ROM                            00000000 00000493     494h

SEGMENTS WITHIN SPACE:
======================

ROM                            Type     Base     Top      Span
```

```
------------------------------ ----------- -------- -------- --------
code                     relocatable 00000000 00000493     494h

  Zilog Linkage Editor.  Version T2.11    16-May-101    18:15:11    Page:   3

SEGMENTS WITHIN MODULES:
============================

Module: main.asm (File: D:\battcharger\main.o) Wed May 16 18:15:07 2001

      Name                          Base     Top      Size
      ------------------------------ -------- -------- --------
Segment: code                       00000000 000000FE     255

Module: KEYNEW.asm (File: D:\battcharger\KEYNEW.o) Mon May 07 10:42:35 2001

      Name                          Base     Top      Size
      ------------------------------ -------- -------- --------
Segment: code                       000000FF 0000023F     321

Module: SRVKEY.asm (File: D:\battcharger\SRVKEY.o) Wed May 16 18:15:09 2001

      Name                          Base     Top      Size
      ------------------------------ -------- -------- --------
Segment: code                       00000240 000002FF     192

Module: Dialout.asm (File: D:\battcharger\Dialout.o) Wed May 16 18:05:57 2001

      Name                          Base     Top      Size
      ------------------------------ -------- -------- --------
Segment: code                       00000300 000003ED     238

Module: UTIL.asm (File: D:\battcharger\UTIL.o) Mon May 07 10:42:40 2001

      Name                          Base     Top      Size
      ------------------------------ -------- -------- --------
Segment: code                       000003EE 0000046B     126

Module: irq.asm (File: D:\battcharger\irq.o) Mon May 07 10:42:42 2001

      Name                          Base     Top      Size
      ------------------------------ -------- -------- --------
Segment: code                       0000046C 00000493     40
```

```
Zilog Linkage Editor.  Version T2.11    16-May-101    18:15:11    Page:   4

EXTERNAL DEFINITIONS BY ADDRESS:
================================

Symbol                          Address  Module       Segment
------------------------------- -------- ------------ -------------------------------
WaitForKeyPress                 000000FF KEYNEW.asm   code
WaitForKeyPressUserDelay        0000011D KEYNEW.asm   code
ScanKeyPad                      0000014D KEYNEW.asm   code
wait_key_off                    000001B1 KEYNEW.asm   code
WaitForKeyRelease               000001B1 KEYNEW.asm   code
WaitForKeyReleaseFlashRed       000001BA KEYNEW.asm   code
delay_10ms                      000001C3 KEYNEW.asm   code
delay_100ms                     000001CF KEYNEW.asm   code
delay_500uS                     000001DB KEYNEW.asm   code
delay_ms                        000001E7 KEYNEW.asm   code
port_delay                      000001F6 KEYNEW.asm   code
check_key                       000001FF KEYNEW.asm   code
WaitForKeyReleaseAndStartThreeSe 00000210 KEYNEW.asm   code
ServiceCode                     00000240 SRVKEY.asm   code
DialOut                         00000300 Dialout.asm  code
set_t16_timer                   000003EE UTIL.asm     code
disable_t16_timer               00000409 UTIL.asm     code
FlashGreenLed                   0000041A UTIL.asm     code
FlashRedLed                     00000432 UTIL.asm     code
mul_8                           0000044D UTIL.asm     code
mult_16                         0000045D UTIL.asm     code
irq3                            0000046C irq.asm      code
irq5                            00000493 irq.asm      code
irq4                            00000493 irq.asm      code
irq2                            00000493 irq.asm      code
irq1                            00000493 irq.asm      code
irq0                            00000493 irq.asm      code
KEY_NUMBER                      0000003C main.asm     (unknown)
KEY_NUMBER_BUFFER               0000003E main.asm     (unknown)

 29 External symbols.
 Zilog Linkage Editor.  Version T2.11    16-May-101    18:15:11    Page:   5

EXTERNAL DEFINITIONS BY NAME:
=============================

Symbol                          Address  Module       Segment
```

```
------------------------------ -------- ------------ ------------------------------
check_key                    000001FF KEYNEW.asm   code
delay_100ms                  000001CF KEYNEW.asm   code
delay_10ms                   000001C3 KEYNEW.asm   code
delay_500uS                  000001DB KEYNEW.asm   code
delay_ms                     000001E7 KEYNEW.asm   code
DialOut                      00000300 Dialout.asm  code
disable_t16_timer            00000409 UTIL.asm     code
FlashGreenLed                0000041A UTIL.asm     code
FlashRedLed                  00000432 UTIL.asm     code
irq0                         00000493 irq.asm      code
irq1                         00000493 irq.asm      code
irq2                         00000493 irq.asm      code
irq3                         0000046C irq.asm      code
irq4                         00000493 irq.asm      code
irq5                         00000493 irq.asm      code
KEY_NUMBER                   0000003C main.asm     (unknown)
KEY_NUMBER_BUFFER            0000003E main.asm     (unknown)
mul_8                        0000044D UTIL.asm     code
mult_16                      0000045D UTIL.asm     code
port_delay                   000001F6 KEYNEW.asm   code
ScanKeyPad                   0000014D KEYNEW.asm   code
ServiceCode                  00000240 SRVKEY.asm   code
set_t16_timer                000003EE UTIL.asm     code
wait_key_off                 000001B1 KEYNEW.asm   code
WaitForKeyPress              000000FF KEYNEW.asm   code
WaitForKeyPressUserDelay     0000011D KEYNEW.asm   code
WaitForKeyRelease            000001B1 KEYNEW.asm   code
WaitForKeyReleaseAndStartThreeSe 00000210 KEYNEW.asm   code
WaitForKeyReleaseFlashRed    000001BA KEYNEW.asm   code

  29 External symbols.
  Zilog Linkage Editor.  Version T2.11   16-May-101   18:15:11   Page:   6

 SYMBOL CROSS REFERENCE:
 ========================

 Symbol                        Module       Use
 ------------------------------ ------------ ----------
 check_key                     KEYNEW.asm   Definition
 delay_100ms                   KEYNEW.asm   Definition
                               main.asm     Reference
                               Dialout.asm  Reference
 delay_10ms                    KEYNEW.asm   Definition
```

	Dialout.asm	Reference
	UTIL.asm	Reference
delay_500uS	KEYNEW.asm	Definition
delay_ms	KEYNEW.asm	Definition
DialOut	Dialout.asm	Definition
	main.asm	Reference
disable_t16_timer	UTIL.asm	Definition
	main.asm	Reference
	KEYNEW.asm	Reference
	Dialout.asm	Reference
FlashGreenLed	UTIL.asm	Definition
FlashRedLed	UTIL.asm	Definition
irq0	irq.asm	Definition
	main.asm	Reference
irq1	irq.asm	Definition
	main.asm	Reference
irq2	irq.asm	Definition
	main.asm	Reference
irq3	irq.asm	Definition
	main.asm	Reference
irq4	irq.asm	Definition
	main.asm	Reference
irq5	irq.asm	Definition
	main.asm	Reference
KEY_NUMBER	main.asm	Definition
KEY_NUMBER_BUFFER	main.asm	Definition
mul_8	UTIL.asm	Definition
	KEYNEW.asm	Reference
mult_16	UTIL.asm	Definition
port_delay	KEYNEW.asm	Definition
	SRVKEY.asm	Reference
	Dialout.asm	Reference
ScanKeyPad	KEYNEW.asm	Definition
ServiceCode	SRVKEY.asm	Definition
	main.asm	Reference
set_t16_timer	UTIL.asm	Definition
	KEYNEW.asm	Reference
	Dialout.asm	Reference
wait_key_off	KEYNEW.asm	Definition
	SRVKEY.asm	Reference
WaitForKeyPress	KEYNEW.asm	Definition
	main.asm	Reference
	SRVKEY.asm	Reference
WaitForKeyPressUserDelay	KEYNEW.asm	Definition

dcbc

```
                    SRVKEY.asm   Reference
WaitForKeyRelease            KEYNEW.asm   Definition
WaitForKeyReleaseAndStartThreeSe KEYNEW.asm   Definition
WaitForKeyReleaseFlashRed       KEYNEW.asm   Definition

End of link map:
  Zilog Linkage Editor.  Version T2.11    16-May-101    18:15:11    Page:   7

Symbol                    Module       Use
------------------------------ ------------ ----------
=================
   0 Warnings
   0 Errors
```

09655374.052601

```
irq5     X 00000493
irq4     X 00000493
irq2     X 00000493
irq1     X 00000493
irq0     X 00000493
irq3     X 0000046C
mult_16  X 0000045D
mul_8    X 0000044D
FlashRedLed X 00000432
FlashGreenLed X 0000041A
disable_t16_timer X 00000409
set_t16_timer X 000003EE
DialOut  X 00000300
ServiceCode X 00000240
WaitForKeyReleaseAndStartThree X 00000210
check_key X 000001FF
port_delay X 000001F6
delay_ms X 000001E7
delay_500uS X 000001DB
delay_100ms X 000001CF
delay_10ms X 000001C3
WaitForKeyReleaseFlashRed X 000001BA
wait_key_off X 000001B1
WaitForKeyRelease X 000001B1
ScanKeyPad X 0000014D
WaitForKeyPressUserDelay X 0000011D
WaitForKeyPress X 000000FF
KEY_NUMBER X 0000003C
KEY_NUMBER_BUFFER X 0000003E
```

```
ZiLOG Developer Studio Workspace File
# WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

[PRJ VERSION]
#begin
VERSION = 300
#end

[MCU TARGET]
#begin
NAME = Z86L72
#end

[EMULATOR]
#begin
NAME = Z86L7100ZEM
#end

[PRJ NAME]
#begin
NAME = dcbc.zws
TYPE = APP
#end

[PRJ PATH]
#begin
PATH = D:\battcharger\
#end

[FILES]
#begin
SOURCE = main.asm
SOURCE = KEYNEW.asm
SOURCE = SRVKEY.asm
SOURCE = Dialout.asm
SOURCE = UTIL.asm
SOURCE = irq.asm
#end

[DEPENDENCIES]
#begin
DEP = fvt.inc
DEP = keydef.inc
DEP = data.inc
DEP = equ.inc
DEP = macro.inc
#end

[C SETTINGS]
#begin
C = -g
C = -Ms
C =  -W
C = -ZiLOG
#end

[ASM SETTINGS]
#begin
ASM =  -l -g -q
#end

[LNK SETTINGS]
#begin
LNK =  -Z -g -m -q
LNK =  -r RFILE %0 : %FF
LNK =  -r XDATA %4000 : %FFFF
LNK =  -r ROM %0 : %3FFF
#end

[LIB SETTINGS]
#begin
LIB = -q
#end

[DEBUG SETTINGS]
#begin
VALUE = 0
PAD = 0
#end

[COM SETTINGS]
#begin
PORT = COM2
BR = 57600
#end

[WND STATUS]
#begin
```

```
ID = 0
OPEN = 0
ID = 1
OPEN = 0
ID = 2
OPEN = 0
ID = 3
OPEN = 0
ID = 4
OPEN = 0
ID = 5
OPEN = 0
ID = 6
OPEN = 0
ID = 7
OPEN = 0
ID = 8
OPEN = 0
ID = 9
OPEN = 0
ID = 10
OPEN = 0
ID = 11
OPEN = 0
ID = 12
OPEN = 0
ID = 13
OPEN = 0
ID = 14
OPEN = 0
ID = 15
OPEN = 0
ID = 16
OPEN = 0
ID = 17
OPEN = 0
ID = 18
OPEN = 0
ID = 19
OPEN = 0
ID = 20
OPEN = 0
ID = 21
OPEN = 0
```

```
ID = 22
OPEN = 0
ID = 23
OPEN = 0
ID = 24
OPEN = 0
ID = 25
OPEN = 0
ID = 26
OPEN = 0
ID = 27
OPEN = 0
#end

[OTP SETTINGS]
#begin
DEVICE = Z86E72
TOPMARK = Standard
TYPE = -2124744304
OPTIONS = 0
METHOD = 0
SIZE = 1
ADDRESS = 4294967295
SERIALNUMBER = 4294967295
REP = 0
#end

[WATCH SETTINGS]
#begin

#end

[C WATCH SETTINGS]
#begin
TabId = 0
TabId = 1
TabId = 2
TabId = 3
#end

[OVERRIDE SETTINGS]
#begin
STATUS = 0
SIZE = 16384
```

```
#end

[BREAKPOINT SETTINGS]
#begin
FILENAME = D:\battcharger\SRVKEY.asm
LINE = 201
ADDRESS = 0x000002fa
FILENAME = D:\battcharger\SRVKEY.asm
LINE = 198
ADDRESS = 0x000002f4
FILENAME = D:\battcharger\SRVKEY.asm
LINE = 112
ADDRESS = 0x0000028b
FILENAME = D:\battcharger\Dialout.asm
LINE = 146
ADDRESS = 0x00000390
FILENAME = D:\battcharger\Dialout.asm
LINE = 138
ADDRESS = 0x0000038c
FILENAME = D:\battcharger\main.asm
LINE = 184
ADDRESS = 0x000000a4
FILENAME = D:\battcharger\main.asm
LINE = 165
ADDRESS = 0x00000098
#end
```

dcbc

```
:00000003FD
:00000001FF
```

```
:10000000049304930493046C04930493E6FE00E6C3
:10001000FFEFE6F700E6F6FFE6F804E6FD0DE60082
:1000200020E60103E60220E6FD0FE60F00E600FEF3
:10003000E60E04760080EB2CE60020E6FD20A63FCD
:10004000AAEB0AA66FAAEB05A67FAA6B17E6FD002E
:1000500058FFB1E500E5A6E505EBF7E63FAAE66F38
:10006000AAE67FAAE6FD20E61100E61200E6020FEE
:10007000E60147460340E600084600FF1803760302
:1000800026B217603046B1CD600FFFBD7D602401F
:10009000D600FF7BF8D604097603026B07760304CB
:1000A0006B028BC0E64CFFD601CFD603008BB54365
:1000B0006F7079726967687420286329323030303034
:1000C0002D3230303120436861D6265726C616938
:1000D0006E2047726F75702E20446576656C6F7068
:1000E000656420627920596 16D7465636820496E8A
:0F00F000632C20383437203936332032383239F8
:1000FF00761102EB08E63201E6310E8B06E632018D
:10010F00E6310ED601B17611026B035600FDE432D3
:10011F0034E43133463002D603EED6014DFB157869
:10012F003C9C32D6014DFB0CA43CE7EB079AF4D66E
:10013F000409DFAF763002EBE1D60409CFAF5600EA
:10014F00F8E63CFFB0E13C01A6E104BB3D48E354B7
:10015F0002E46B071E90E3FBEF8B2FB0E04600FF2E
:10016F005600FEA6E004BB22D601F6D601F65802D1
:10017F0056E50FA6E50FEB17580046E5F890E54654
:10018F00E5F854E500FB030E8BD95600F8CFAFB856
:10019F00E1DC03D6044D02D0D93C5600F8D602213B
:1001AF00DFAF5600F8D601FF7BFBAF5600F8D60144
:1001BF00FF7BFBAF70E33C14D601DB3AFB50E3AFA0
:1001CF0070E44C32D601C34AFB50E4AF70E33C17E6
:1001DF00D601F63AFB50E3AF5600FDD601DB4600E1
:1001EF0002D601DB3AF2AF70E470E450E450E4AFB2
:1001FF005600F8080256E00FA6E00F6B02DFAFCFF4
:10020F00AF0CFF00E06B09D601C3D601FF7BF4AF43
:10021F00DFAF0C021C34043CE116E000C220293C85
:10022F00AFE63CFFAF0102030405060708090A0009
:01023F000CB2
:10024000A63C0CBB0CA63C0A6D02C6A63C0C6D027B
:1002500052AF4611025601BFE64041D601B1E63227
:1002600003E631A9D6011DFD029FD602B07D029F93
:10027000F53C402040D600FFFD029F5600FDA63C05
:100280009BD029FF53C4020408AEAE740FF460056
:10029000FFE61100CF460140461202D601B1AFE79A
:1002A00040FF4600FF460140E611005612FDDFAF59
```

```
:1002B0008C0AA63C006B09A63C016B028C06CFAFF2
:1002C0008C00DF8B20AFD601B1E63203E631A9E620
:1002D0000106D601F6D6011DFD02DEE60107E601A4
:1002E00007461100AFD601B1E60103D601F6E632AA
:1002F00003E631A9D6011DFD02DEE60107DF8BDE34
:10030000E60187D603B5E64041460180D601CFD647
:1003100001CFE540E2A6E2FF6B15D601CFD603D5AB
:100320002900E60320E60340460008204O8BE3E670
:100330000360D601CFD601CFE63401E6330E463056
:1003400002D603EED601F6763002EBFBD601F6E6D6
:100350003403E633A9463002D603EE5601FBD6013C
:10036000F6763002EBFBD60409E60187D601F6E605
:100370003403E633A9463002D603EEE60183D60104
:10038000F6763002EBFBD60409E601F756017FAFA3
:100390005600F7D601F65603EF4603205603BFAFCB
:1003A000E6F8445603BF46000846031O5603DFD65E
:1003B00001C3FFFFAFE60350E60019E60320E603A2
:1003C00050E60099E60320E60350E60008E6032025
:1003D000E60350AFAF0C031CE4021216E000C2208B
:0E03E000AF2CFFAFA0112031405160718091 11
:1003EE0070FDE6FD2DE606FFE607FFE60226E601B6
:1003FE00F346028046FB089F50FDAF70FDE6FD2DD3
:10040E00E6022056FBF750FD5630FDAF70E88C0A21
:10041E005600FDD601C3460002D601C3D601C38ADB
:10042E00EF50E8AF70E88C05560000D601C3D60138
:10043E00C34600FFD601C3D601C38AEC50E8AFEC29
:10044E0009B0ECCFC0ECC0EDFB0202CBEAF6AFC8B0
:0E045E00EAD8EB00E96B0602BD12AC9AFAAFC9
:10046C0070FDE6FD3D7602206B1A46022076E00216
:10047C006B1226E30136E400EB0AA6E300EB05560B
:08048C00E0FD8B0050FDBFBF35
:00000003FD
:00000001FF
```

```
;*************************************************************************
;
; FILENAME:  findkey.s


;


;


;     DESCRIPTION:


;


;     REVISION HISTORY:

          V1.0   Date:    6/7/94
          V2.0   Date:    8/14/96
          V3.0   Date:    10/96       Author:



;*************************************************************************



;** include files **


      .include  fvt.h
      .include  data.h
      .include  equ.h
      .include  keydef.h
      .include  macro.h

;** external functions **

      .extern ParseDAT
      .extern CheckPunchThru
```

```
        .extern WaitForKeyReleaseFlashRed
        .extern SendIR
        .extern SetDriver
        .extern ScanKeyMap

        .extern mul_8
        .extern check_key
        .extern blink_green

        .extern GetSleepTime
        .extern ConfigForSleep
        .extern DeviceLightsOff

        .extern CheckVolPriority

;** public functions **

        .global SendCode

;*************************************************************
;       FILENAME:  findkey.s
;
;  SendCode
;
;       Version:
;       Date: 09/30/96, 11:42:30
;       Author:
;
;       Function: Sends Key Data for all send modes, ie Normal, Action, and Double Actio
n
;
;       Inputs:
;               KEY_NUMBER = (range of 0 to 36), only sending keys if it got here
;               IR_MODE = signifies if in action/double action/punchthru/ ....
;
;               SELECTED_DEVICE = last device key hit /// ACTIVE_DEVICE = current
device to send
;               DEVICE_FLAG = last device key hit to restore last code after punchthru
;               SendCounter = number of times to send 1 = send once (if 0 then send once

only)
;               OVERLAY_ADDR, OVERLAY_ADDR+1: address to overlayed dat file
;
MapSize     .equ    #40
```

```
;
;	Returns:
;		CF = 0  - OK
;		CF = 1  - Error
;
;	Modifies:
;
;	Subords:
;
;*******************************************************************
SendCode:
	;** punch thru overhead **

	push	DEVICE_FLAG				; in case we do
punch thru we can get back to original mode

	ld	ACTIVE_DEVICE, SELECTED_DEVICE
	ld	ACTIVE_DEVICE+1, SELECTED_DEVICE+1

	;** volume priority **
	call	CheckVolPriority			; reconfigs for device with
volme priority

PunchThruStart:
	and	%fc, #11111100b				; clear user flags

	ld	r0, ACTIVE_DEVICE
	or	r0, ACTIVE_DEVICE+1
	jp	z, ExitError				; code isn't programmed,
ExitError

	ld	r0, ACTIVE_DEVICE
	ld	r1, ACTIVE_DEVICE+1			; rr0 = pointer to
dat file

	;** rr0 ptr to dat file, parse the 1st and 2nd byte of the dat file **
	ldc	r3, @rr0
	ld	MOD_TYPE, r3
	incw	rr0
	ldc	r3, @rr0
	ld	CODE_LEN_BITS, r3			; CODE_LEN_BITS defined

	and	r3, #3fh
	ld	r4, r3
```

```
     sra    r3
     sra    r3                    ; divide
CODE_LEN_BITS by 8
     sra    r3
     and    r4, #07h
     jr     z, NoRemainder              ; if the lower 3
bits are 0, there will be no remainder.
     inc    r3
NoRemainder:
     ld     CODE_LEN_BYTES, r3          ;
CODE_LEN_BYTES defined

     ;** parse the 1st and 2nd byte of the dat file **
     incw   rr0                   ; point to
CARRIER (3rd byte of dat file)
     call   ParseDAT                    ; inputs: rr0=ptr to dat+2
(CARRIER), MOD_TYPE; output:rr0 points to keymap[0]

     ;** adjust KEY_NUMBER to action/double_action range **
     tbitnz IR_MODE, #DOUBLE_ACTION_MODE, InDoubleActionMode
     tbitnz IR_MODE, #ACTION_MODE, InActionMode
     jr     OverHeadDone

InDoubleActionMode:
     add    KEY_NUMBER, #MapSize              ; adjust to
ActionMode
InActionMode:
     add    KEY_NUMBER, #MapSize              ; adjust to
ActionMode

OverHeadDone:
     ld     KEY_NUMBER_BUFFER, KEY_NUMBER          ; make a copy for
checkpowerflag
     ;** KEY_NUMBER defined, check if pip special feature **
;    call   CheckPipFlag                ; add 40 to
KEY_NUMBER if needed

     ;** rr0 points byte after FLAG byte! Either keymap[0] or standardkeygroup **
     call   CheckStandardKeys               ; adjust rr0 to
STANDARD_NUMBER_GROUP if FLAG set
     jr     nc, OffsetConfigured            ; c=0 if standard key

     ;** rr0 points to keymap [0] **
     call   ScanKeyMap                  ; input:
```

```
KEY_NUBMER and rr0 = pointer to keymap[0]; output: KEY_NUMBER = offset into dat
      jr      nc, KeyInMap

      ;** not in keymap, check if it's overlayed **
      call    CheckOverlay                    ; returns: rr0
points to flag byte
      jr      c, NoOverlay
      call    ReconfigFlagByte                  ; inputs: rr0 points to flag byt
e
      incw    rr0                         ; point to byte
after flag byte
      jr      OverHeadDone

NoOverlay:
      call    CheckPunchThru                    ; returns: c=0 if
punchthru
      jp      nc, PunchThruStart
      jr      ExitError

;*************************************************************
;rr0 points to rawkeydata[0]
;KEY_NUMBER is to offset into rawkeydata[table]
;
;*************************************************************
KeyInMap:
      call    CheckPowerFlag                    ; adjust
KEY_NUMBER if [POWER] key hit

OffsetConfigured:
      ;** rr0 and KEY_NUMBER configured - move rr0 to rawkeydata **
      ld      r13, CODE_LEN_BYTES
      ld      r11, KEY_NUMBER                    ;holds the actual
key number
      call    mul_8
      addw    r0,r1,#0,r13

      cp      RF_TOGGLE, #0ffh
      jr      ne, NoRF
      ;RFOn
NoRF:
;     tbitnz  IR_MODE, #SCAN_MODE, NoDevLights       ;in scan mode don't mess with
the dev lights
      call    DeviceLightsOn                   ;this line must
precede the call to set driver
```

```
;NoDevLights:
      ;driver to reconstruct the code in Ram
      and   CODE_LEN_BITS,#3fh
      call  SetDriver
      jr    c, DriverIsSelfContained

      call  SendIR

DriverIsSelfContained:
      ;RFOff
      pop   DEVICE_FLAG
      rcf
      ret                             ; Exit point for IR
transmission

ExitError:
;     tbitnz IR_MODE, #SCAN_MODE, NoLightsScan     ;in scan mode don't mess with
the dev lights
      call  DeviceLightsOn

;NoLightsScan:
      pop   DEVICE_FLAG                      ; in case we do
punch thru we can get back to origonal mode
      call  WaitForKeyReleaseFlashRed
      scf
      ret

;*****************************************************************
;     FILENAME:  findkey.s
;
; CheckPipFlag
;
;     Version:
;     Date: 11/01/96, 13:59:26
;     Author:
;
;     Function: For picture in picture devices
;            If pip key hit then menu cluster goes into pip mode.
;            If menu key hit then menu cluster goes into menu mode.
;            If in double action then xx_pip == pip
;
;
```

```
;     Inputs: KEY_NUMBER
;           IR_MODE (pip flag)
;
;     Returns:KEY_NUMBER = KEY_NUMBER+40 if flag and mode and key hit
;           IR_MODE (pip flag - set or clears)
;
;***********************************************************
CheckPipFlag:
PipExit:ret




;***********************************************************
;     FILENAME:  findkey.s
;
;  ReconfigFlagByte
;
;     Version:
;     Date: 10/31/96, 10:16:49
;     Author:
;
;     Function: For overlay codes when re-reading the master dat file reconfig the FLG
;S byte
;            to correctly grap the correct key data.
;
;
;     Inputs: rr0 points to flag byte in dat file
;           FLGS = FLGS for the overlayed code
;     Returns:
;
;     Modifies:
;
;     Subords:
;
;***********************************************************
ReconfigFlagByte:
      ;** rr0 points to flag byte **
      ldc    r2, @rr0                      ; FlagByte

      ;** reconfig std flag **
      tbitz  r2,#STANDARD_KEY_FLAG, FlagNotSet0
      or     FLGS, #STANDARD_KEY_FLAG             ; set flag
```

```
    jr     DoAction
FlagNotSet0:
    and    FLGS, #^c STANDARD_KEY_FLAG          ; clear flag

DoAction:
    ;** reconfig action flag **
    tbitz  r2,#ACTION_FLAG, FlagNotSet1
    or     FLGS, #ACTION_FLAG              ; set flag
    jr     DoDoubleAction
FlagNotSet1:
    and    FLGS, #^c ACTION_FLAG              ; clear flag

DoDoubleAction:
    ;** reconfig double action flag **
    tbitz  r2,#DOUBLE_ACTION_FLAG, FlagNotSet2
    or     FLGS, #DOUBLE_ACTION_FLAG           ; set flag
    jr     Exit2
FlagNotSet2:
    and    FLGS, #^c DOUBLE_ACTION_FLAG         ; clear flag

Exit2:  ret

;*************************************************************
;
;    FILENAME:  findkey.s
;
;  CheckOverlay
;
;    Version:
;    Date: 10/02/96, 13:41:43
;    Author:
;
;    Function:
;
;    Inputs: CODE_LEN_BITS, BIT #40h if set then overlayed code
;         OVERLAY_ADDR, OVERLAY_ADDR+1: address to overlayed dat file
;
;    Returns:
;         CF = 0  - Overlayed (also clears overlay flag)
;         CF = 1  - NotOverlayed or Overlayed Checked Once already or in
Action_Mode
;
;*************************************************************
CheckOverlay:
        tbitz  CODE_LEN_BITS, #OVERLAY_FLG, NotOverlayed

        and    CODE_LEN_BITS, #^c OVERLAY_FLG        ; clear overlay flg.
        ld     r0, OVERLAY_ADDR
        ld     r1, OVERLAY_ADDR+1
        rcf
        ret

NotOverlayed:
        scf
        ret


;*********************************************************
;       FILENAME:  findkey.s
;
;CheckPowerFlag
;
;       Version:
;       Date: 10/01/96, 14:36:39
;       Author:
;
;       Function: Determines if 1 is added to KEY_NUMBER.
;               Will add 1 if POWER_FLG set and:
;                               if toggle=power off
;                               or if in Action modes
(key_number > action_0)
;
;       Inputs: KEY_NUMBER
;           rr0 = POWER key data
;           KEY_NUMBER = offset into dat file (1byte/key)
;           KEY_NUMBER_BUFFER = key hit
;
;       Returns:
;           rr0 = points to POWER_ON or POWER_OFF key data (depending on
toggle)
;           KEY_NUMBER = offset into dat file raw data
;
;*********************************************************
CheckPowerFlag:
        tbitz  CODE_LEN_BITS, #POWER_FLG, cpf_exit
        cp     KEY_NUMBER_BUFFER, #X_KEY_0
        jr     uge, AddOne
```

```
    cp    KEY_NUMBER_BUFFER, #KEY_POWER
    jr    nz, cpf_exit

    inc   POWER_SEND_FLG
    tbitnz POWER_SEND_FLG, #BIT0, cpf_exit
AddOne:
    add   KEY_NUMBER, #1                ; send power_off
if POWER_SEND_FLG = odd
cpf_exit:
    ret

;*****************************************************************
;    FILENAME:  findkey.s
;
; CheckStandardKeys
;
;    Version:
;    Date: 10/01/96, 16:13:33
;    Author:
;
;    Function:
;
;
;
;
;    Inputs: rr0 = byte after the flag byte: maybe keymap[0] or stand_num_group
;         KEY_NUMBER = offset to raw key in dat file
;
;    Returns:
;         if standard group:
;              rr0 = pointer to Standard_Number_Group if digit hit
;              CF = 0  - number key hit and STANDARD_KEY_FLAG set
;         else
;              rr0 = pointer to keymap[0]
;              CF = 1  - not a number key
;
;
;
;*****************************************************************
CheckStandardKeys:
    tbitz  FLGS,#STANDARD_KEY_FLAG, FlagNotSet
    cp     KEY_NUMBER, #9
    jr     ugt,NotANumberKey

    ;number key hit and STANDARD_KEY_FLAG set
```

```
        ldc     r2,@rr0                         ; standard key
set# defined

        rcf
        rlc     r2
        ldrr    r8,r9,STANDARD_KEY_TABLE
        addw    r8,r9,#0,r2
        ldc     r0,@rr8
        incw    rr8
        ldc     r1,@rr8

        rcf
        ret

NotANumberKey:
        incw    rr0                             ; point to
KeyMap[0]

FlagNotSet:
        scf
        ret


;**********************************************************
;       FILENAME: g:\500\540\findkey.s
;
;  DeviceLightsOn
;
;       Version:
;       Date: 12/09/96, 11:22:03
;       Author:
;
;       Function:
;
;
;
;       Inputs:
;
;       Returns:
;               CF = 0  - OK
;               CF = 1  - Error
;
;       Modifies:
;
```

```
;       Subords:
;
;
;*********************************************************
DeviceLightsOn:               ;lights up the Active Device key
        call   DeviceLightsOff

        push   DEVICE_FLAG          ;Save it

        cp     DEVICE_FLAG,#0b0h
        jr     ult,Sat

        sub    DEVICE_FLAG,#70h
        RedLedOn
Sat:
        cp     DEVICE_FLAG, #DEV_SAT
        jr     ne, NotSat

        SatLedOn
        ;ret

NotSat:
        cp     DEVICE_FLAG, #DEV_VCR
        jr     ne, NotVcr

        VcrLedOn
        ;ret

NotVcr:
        cp     DEVICE_FLAG, #DEV_TV
        jr     ne, NotTV

        TvLedOn
        ;ret

NotTV:
        cp     DEVICE_FLAG, #DEV_RCVR
        jr     ne, NotCbl

        CblAmpLedOn
        ;ret
NotCbl:
        cp     DEVICE_FLAG,#DEV_CABLE
        jr     ne,Ledret
        AuxLedOn
```

```
Ledret:
        pop     DEVICE_FLAG
        ret


STANDARD_KEY_TABLE:
        .extern SET_0,SET_1,SET_2,SET_3,SET_4,SET_5,SET_6,SET_7
        .extern SET_8,SET_9,SET_10,SET_11,SET_12,SET_13,SET_14
        .extern SET_15,SET_16,SET_17,SET_18,SET_19,SET_20,SET_21,SET_22
        .extern SET_23,SET_24,SET_25,SET_26,SET_27,SET_28,SET_29
        .extern SET_30,SET_31,SET_32,SET_33,SET_34,SET_35,SET_36,SET_37
        .extern SET_38,SET_39,SET_40,SET_41,SET_42,SET_43,SET_44
        .extern SET_45,SET_46,SET_47,SET_48,SET_49,SET_50,SET_51
        .extern SET_52,SET_53,SET_54,SET_55,SET_56,SET_57,SET_58,SET_59

        .word   SET_0
        .word   SET_1
        .word   SET_2
        .word   SET_3
        .word   SET_4
        .word   SET_5
        .word   SET_6
        .word   SET_7
        .word   SET_8
        .word   SET_9
        .word   SET_10
        .word   SET_11
        .word   SET_12
        .word   SET_13
        .word   SET_14
        .word   SET_15
        .word   SET_16
        .word   SET_17
        .word   SET_18
        .word   SET_19
        .word   SET_20
        .word   SET_21
        .word   SET_22
        .word   SET_23
        .word   SET_24
        .word   SET_25
        .word   SET_26
        .word   SET_27
```

```
.word   SET_28
.word   SET_29
.word   SET_30
.word   SET_31
.word   SET_32
.word   SET_33
.word   SET_34
.word   SET_35
.word   SET_36
.word   SET_37
.word   SET_38
.word   SET_39
.word   SET_40
.word   SET_41
.word   SET_42
.word   SET_43
.word   SET_44
.word   SET_45
.word   SET_46
.word   SET_47
.word   SET_48
.word   SET_49
.word   SET_40
.word   SET_51
.word   SET_52
.word   SET_53
.word   SET_54
.word   SET_55
.word   SET_56
.word   SET_57
.word   SET_58
.word   SET_59
.end
```